- MANUAL TÉCNICO -

Software de diagnóstico de la seguridad de la información y autoimplantación de LOPD

Rev. 01- FEBRERO 2013

Software de diagnóstico de la seguridad de la información y autoimplantación de LOPD

Teléfono Afogasca: 945 286 962

Email Afonvi: afogasca@afogasca.es

REALIZADO POR: TEKAGIS S.L.

Índice

CONSIDERACIONES DEL DISEÑO....3
Perfiles....3
Requisitos....3
ÁRBOL DE DIRECTORIOS....6
Estructura de directorios....6
index.php....6
assets....6
protected....6
protected>components....6
protected>config....6
protected>controllers....7
protected>extensions....7
protected>models....7
protected>views....7
themes>afogasca....7
Núcleo de Yii y librerías....7
DISEÑO DE LA APLICACIÓN....9
Arquitectura general del sistema....9
MODELOS, CONTROLADORES, VISTAS Y API....12
API....12
CONEXIÓN CON AEPD Y NOTIFICACIÓN DE FICHEROS....12

CONSIDERACIONES DEL DISEÑO

Perfiles

Esta aplicación cuenta únicamente un tipo de usuario (la empresa asociada) que puede usar la aplicación enteramente con ese perfil.

Cada usuario dispone de su propia base de datos.

Requisitos

Rf1: Cada usuario tiene sus datos en una base de datos independiente del resto.
Rf2: Existe una base de datos de control que gestiona las altas de asociados y en la que se consulta a qué base de datos debe acceder cada empresa asociada.
Rf3: El administrador de la base de datos introduce en la base de datos de control los datos de las empresas que pueden darse de alta en la aplicación.
Rf4: Los asociados pueden registrarse desde la aplicación.
Rf5: Se manda un email con los datos del registro.
Rf6: Tiene que existir la opción de recuperar la contraseña enviando un email a la cuenta de la empresa asociada.
Rf7: Dentro de la aplicación debe existir un apartado para modificar la contraseña.
Rf8: Existe un apartado de avisos con los documentos que no están registrados.
Rf9: La aplicación incluye un test inicial que comprueba el grado de seguridad establecido en la empresa.
Rf10: Existe un apartado para introducir los datos de la empresa.

Rf11: Una empresa puede tener más de una sede.
Rf12: Una empresa puede tener más de un encargado de tratamiento.
Rf13: Una empresa puede tener muchos ficheros.
Rf14: Dependiendo de si se ha registrado o modificado pueden cambiar de estado.
Rf15: La aplicación debe calcular el nivel de seguridad que debe mantener la empresa dependiendo del tipo de ficheros que trata. El nivel siempre coincide con el nivel de seguridad más alto de cualquiera de los ficheros que posee.
Rf16: Debe existir una opción para indicar el tipo de seguridad que debe tener la empresa en relación a los datos que trata de terceros.
Rf17: Una empresa puede tener varios responsables de seguridad.
Rf18: Una empresa puede tener varios usuarios de programas informáticos en su organización.
Rf19: Los usuarios pueden tener más de un tipo de perfil distinto.
Rf20: Las empresas pueden tener varias aplicaciones o software.
Rf21: Además, pueden tener varios equipos informáticos y utilizar distintos sistemas operativos.
Rf22: La empresa puede registrar el software de seguridad que utilice en su empresa.
Rf23: Una empresa puede realizar varias copias de seguridad de distintos ficheros.
Rf24: Una empresa puede disponer un inventario de soportes de información que contiene los ficheros de la misma.
Rf25: Las entradas y salidas de información deben poder registrarse en la aplicación.
Rf26: Un asociado puede indicar cualquier tipo de incidencia de seguridad que se produzca en su organización.

Rf27: Se deben poder registrar las peticiones ARCO necesarias para cumplir con la LOPD.
Rf28: La empresa puede registrar las cesiones de datos a terceras partes.
Rf29: La aplicación debe ser capaz de generar los documentos de seguridad a través de la información introducida en el sistema.
Rf30: La aplicación debe incluir un manual de usuario.
Rf31: La aplicación debe estar bajo la licencia GPL.
Rf31: La aplicación debe incluir los términos de privacidad a los que está sujeta la aplicación y los usuarios.
Rf31: La aplicación debe incluir la información de contacto necesaria para que los usuarios puedan contactar si encuentran problemas.

ÁRBOL DE DIRECTORIOS

Estructura de directorios

La aplicación está organizada según la estructura básica elegida por el framework Yii, de la siguiente manera:

index.php

Página principal de la aplicación que carga las configuraciones e inicia la aplicación.

assets

Esta carpeta funciona como caché para cada equipo, por lo tanto tiene que tener permisos de escritura y puede ser vaciada si se traslada a otro equipo.

protected

Esta carpeta incluye toda la funcionalidad de la aplicación.

protected>components

En esta carpeta se incluyen los componentes necesarios para la ejecución de la aplicación.

protected>config

main.php

Este archivo incluye los datos de personalización necesario para adaptar la aplicación a diferentes servidores y situaciones.

protected>controllers

Esta sección incluye el controlador principal (SiteController.php) y los subcontroladores.

protected>extensions

Esta sección incluye las extensiones de comportamiento necesarias para el buen funcionamiento de la aplicación.

protected>models

Esta sección incluye los modelos de datos utilizados y basados en las tablas de la base de datos.

protected>views

Las vistas de listados y formularios se encuentran en este apartado.

themes>afogasca

Los recursos estilísticos se encuentran en esta carpeta, incluye: imágenes, css y descripción de la interfaz en forma de "layouts".

Núcleo de Yii y librerías

Las librerías se encuentran en la carpeta que incluye el núcleo del framework Yii (extensions), esta carpeta no debe ser accedida por usuarios web y por lo tanto se encuentra en un nivel superior.

EExcelView

Extensión de Yii para exportar datos a formatos de hojas de cálculo.

DateTimei18NBehavior

Extensión de Yii que convierte las fechas automáticamente según el idioma establecido.

mpdf54

Librería de para la creación dinámica de PDF necesaria para la obtención de los informes.

PHPMailer

Librería PHP para enviar emails.

DISEÑO DE LA APLICACIÓN

Arquitectura general del sistema

Con los requisitos establecidos por la empresa se decidió seguir una arquitectura que combinara los siguientes estilos arquitectónicos:

Patrón en 3 capas:
El objetivo de este estilo de programación es separar en tres capas la aplicación de modo que los cambios en una capa afecten lo menos posible a las otras.

La **capa de presentación** es la interfaz que se le muestra al usuario y en la que éste puede interactuar con el sistema.

La **capa de negocio** establece la lógica del programa. Recibe las peticiones del usuario y se comunica con el nivel de datos para que le ofrezca los datos que necesita para enviarlos como respuesta a la capa de presentación.

Por último la **capa de datos** es la encargada de obtener los datos del sistema de almacenamiento elegido (distintos SGBDs, ficheros,...) y los entrega a la capa de negocio.

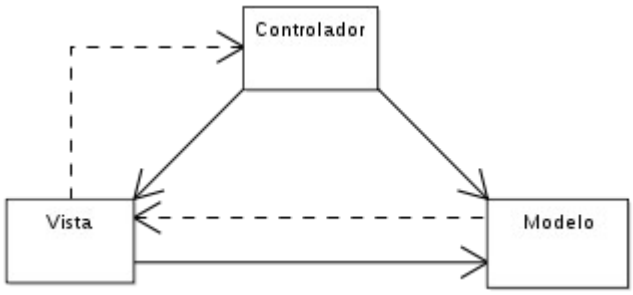
Patrón OO:
Siguiendo este patrón las entidades del mundo real tiene su correspondiente "objeto" en el sistema. Según Wikipedia los objetos son:

"*entidades que combinan estado (atributo), comportamiento (método) e identidad:*

- El estado está compuesto de datos, será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
- El comportamiento está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.
- La identidad es una propiedad de un objeto que lo diferencia del resto"

Patrón Modelo -Vista-Controlador
Para ayudar todavía más a la separación entre las capas se utiliza este patrón que crea una nueva capa intermedia cuya responsabilidad es manejar la comunicación entre vista y modelo de manera que evite lo máximo posible

la programación en la capa de presentación. Este nuevo nivel y relaciones pueden verse en la figura.



Su funcionamiento según Wikipedia es el siguiente:

"Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo

notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.

5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.”

Es posible encontrar más documentación sobre el framework de Yii en la página oficial: http://www.yiiframework.com/doc/

MODELOS, CONTROLADORES, VISTAS Y API

API

Puedes revisar la API con los modelos, controladores y vistas en la carpeta API.

Ten en cuenta, que se encuentran tanto los archivos generados para esta aplicación como aquellas clases y funciones del núcleo de Yii necesarias para la ejecución de la aplicación.

CONEXIÓN CON AEPD Y NOTIFICACIÓN DE FICHEROS

Para notificar el alta de los ficheros de titularidad privada registrados en la aplicación se utiliza el estándar de XML indicado por la AEPD y la conexión al Web Service de la AEPD. Estos documentos están incluidos en la documentación técnica en la carpeta AEPD.